1.Giriş

Bu doküman Programlama Laboratuvarı 1 dersi 2. Projesi için çözümü açıklamaya yönelik oluşturulmuştur. Dökümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projeyi hazırlarken kullanılan kaynaklar bulunmaktadır

2.Temel Bilgiler

Program Visual Studio üzerinde C# program dili ile yazılmıştır.

Görsel olarak bir labirent tasarlanmamıştır. Masaüstü uygulaması olarak oluşturulmuştur.

3.Proje Tanımı

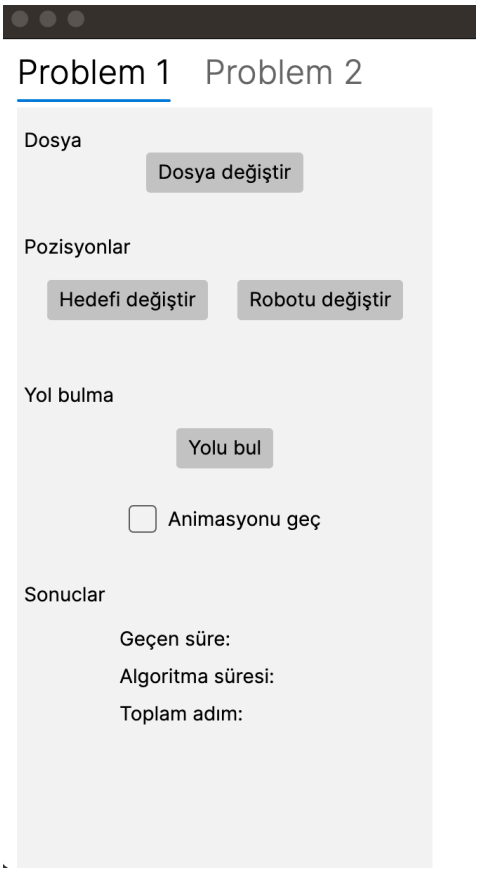
Belirli kurallara göre hareket eden bir robotun önündeki engelleri aşarak istenen hedefe ulaşmasını sağlayan bir oyun tasarlanması beklenmektedir. Oyunda iki adet problemin çözülmesi gerekmektedir.

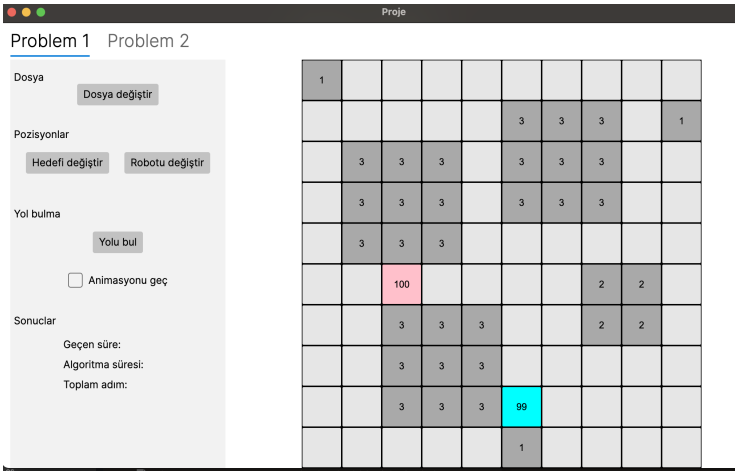
3.1.Proje İsterler

•Bu problemde sizden robotu ızgara (grid) üzerinde verilen hedefe engellere takılmadan en kısa sürede ve en kısa yoldan ulaştırmanız beklenmektedir.

• Gerekli boyutlarda karesel bir ızgara alanı oluşturmanız gerekmektedir.
• Izgara üzerine engeller ve duvarlar yerleştirilmelidir. Izgara boyutu, engel sayısı ve engellerin konum bilgileri içeriği matris biçimindeki bir text dosyasından alınacaktır.
•Robotun başlangıç ve hedef noktaları ızgara üzerindeki uygun (engel veya duvar içermeyen) karelere rastgele belirlenmelidir. Robot başlangıçta tüm ızgara dünyasını bilmemelidir, sadece bir adım sonraki kareleri görebilmelidir.
•Tüm bu bilgiler doğrultusunda, robotun hedefe en kısa sürede ulaşabileceği en kısa yol, adım adım ızgara üzerinde gösterilmelidir.

Arayüzler hakkında birkaç görsel;





•Encapsulation: Sınıflardaki degerler dışarıdan doğrudan erişilmesini engellemek amaciyla protected yapilmistir, erisim icin getter ve setterler kullanilmak zorundadir.
•Inheritance: Bu projede bir GridTemel classımız var ve bu class IntGrid class'ı için bir virtual void olan HucreleriOlustur() metodunu sağlıyor.

•Polymorphism: Overload'a ihtiyac duymadigi icin kullanılmamıştır ama inheritance kisimlarinda override edilen metotlar bulunmaktadir. Projede Pathfinding class'ı sadece IGrid adlı interface'i alıyor, ve bu sayede IGrid olan herhangi bir grid pathfinding class'ı için işe yarayacaktır.

•Abstraction: GridTemel ve IntGrid class'ı abstract olduğu için sadece diğer classlar tarafından inherit edildiği sürece kullanılabilir.

3.2 Yapılan Örnek Fonksiyonlar ve Bazı Source Code Files

GridTemel.cs :Avalonia UI UserControl'in subclass'ı olan bir Grid implementasyonu IGrid interface'ini kullanıyor. Gridi oluşturmak için gerekli bilgileri içeriyor.

Hücre.cs: Grid üzerindeki her bir kutu şeklindeki alanı temsil ediyor. Renk hücretipi, hangi satırda ve sutunda bulunduğu gibi verileri içeriyor.

HucreRenkleri: Düzen açısından static olarak renkleri bu classta ayrı olarak kullanıldı.

LabirentRobot:T değeri LabirentGrid alan bir Subclass. Robot classında bulunan Robot ve hedef yerini değiştirme metodlarını override ederek sadece oluşturulan labirentin giriş ve çıkış değerlerini çizmesini sağlanıyor.

Pathfinding.cs: Yol bulma için gerekli olan algoritmaları içeriyor. Ve bu algoritmalara Algoritma enum'u sayesinde ve bir metodla ulaşabiliyor.

Örnek Fonksiyon

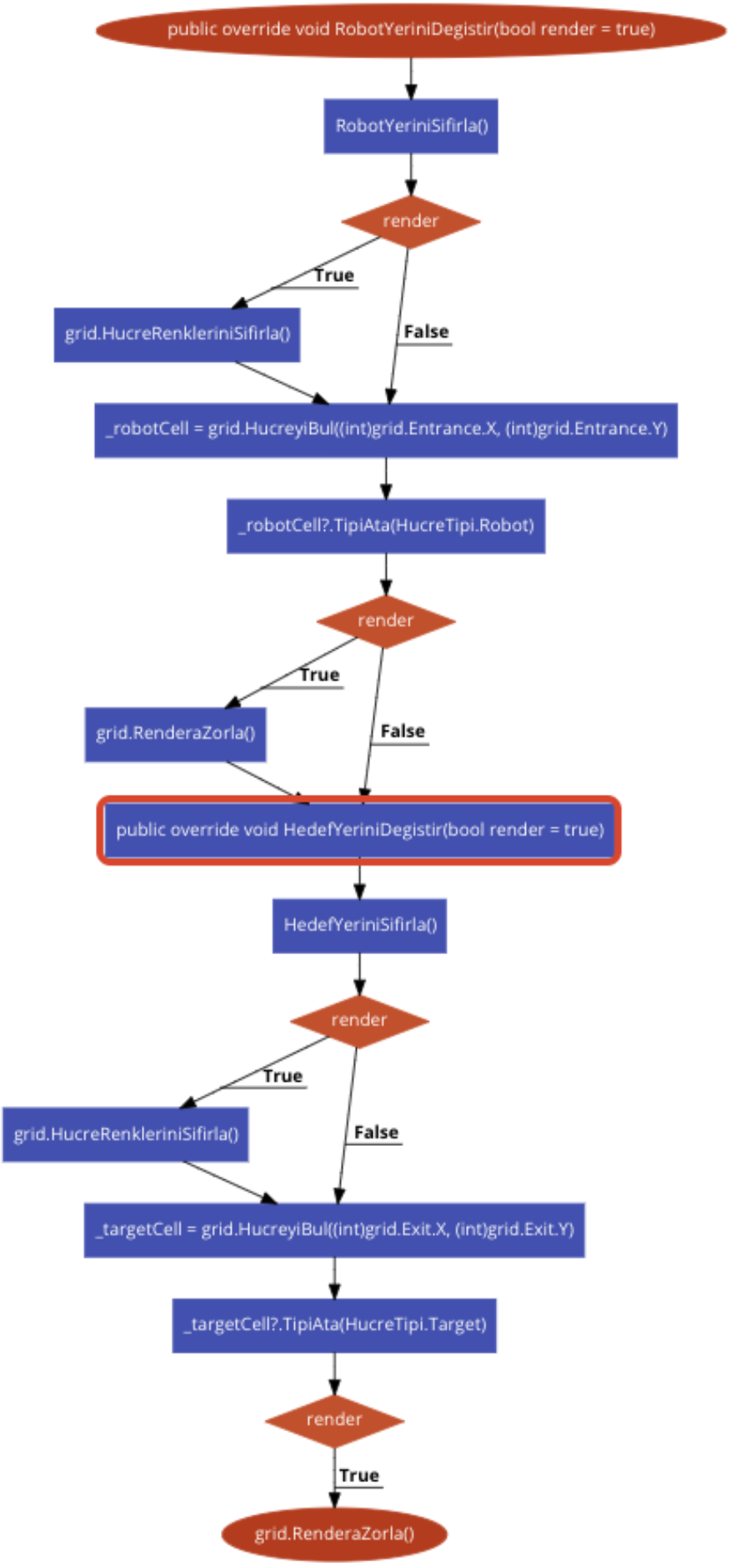
```
private void LabirentGridOlustur(object sender, RoutedEventArgs e)
{
    // Genişlik ve yüksekliği al
    int genislik = (int)labirentGenislik.Value;
    int yukseklik = (int)labirentYukseklik.Value;

    //Yeni labirent oluştur
    labirentGrid.CreateMaze(genislik, yukseklik);
}
```

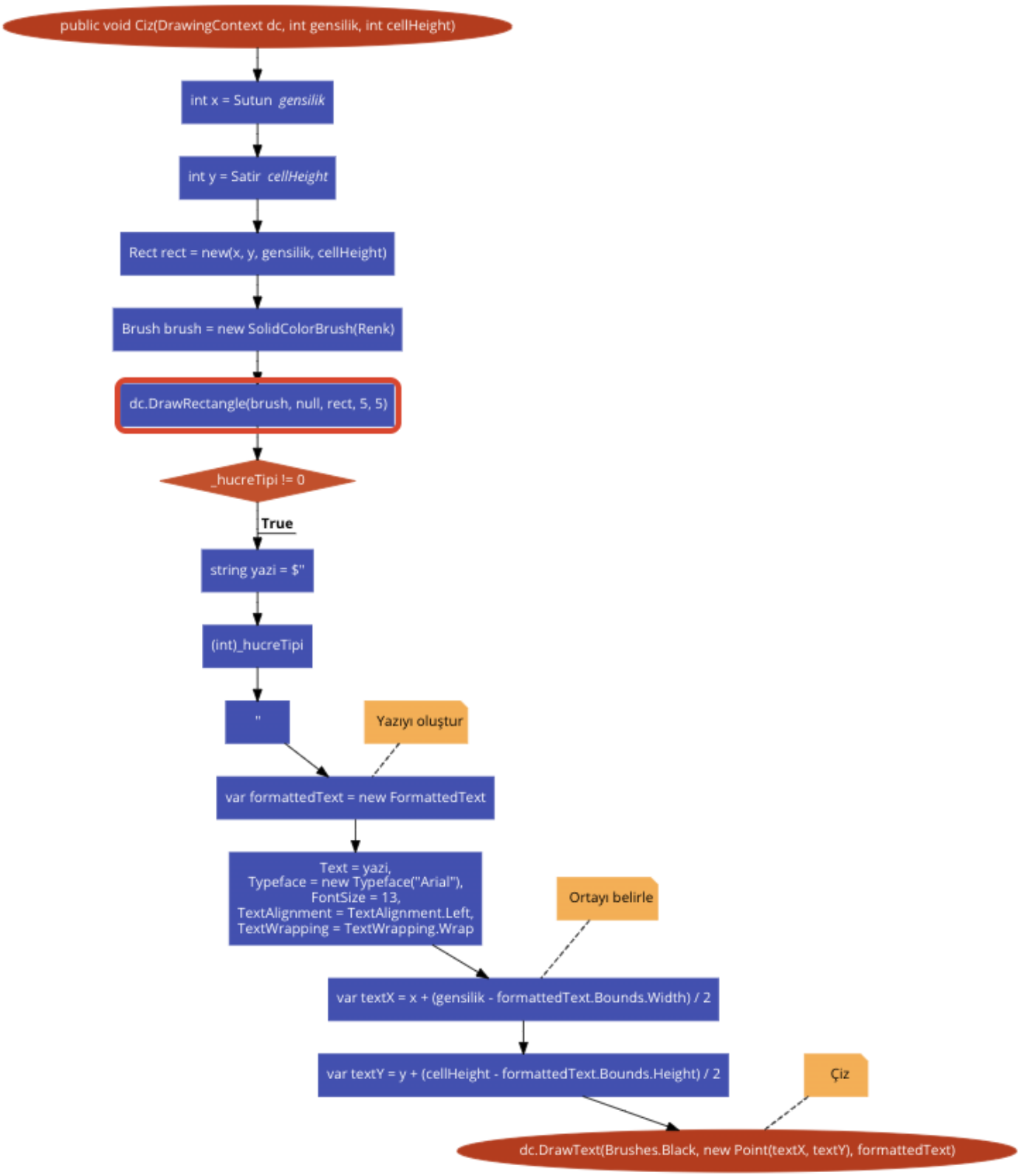
```
public static int[,] Olustur(int genislik, int yukseklik, out Point giris,out Point ciki
{
    // Eğer genislik veya yükseklik çift ise algoritma bozuluyor. O yüzden 1 artır.
    if(genislik % 2 == 0) { genislik++; }
    if(yukseklik % 2 == 0) { yukseklik++; }

    int[,] labirent = new int[yukseklik, genislik];

    // Her hücreyi duvar yap
    for (int i = 0; i < yukseklik; i++)
    {
        for (int j = 0; j < genislik; j++)
        {
            labirent[i, j] = 1;
        }
    }
    // Giriş köşesini belirler
    (int girisSatır, int girisSutun) = random.Next(2) == 0 ? (1, 0) : (0, 1);
    giris = new Point(girisSutun, girisSatır);
```

4. Kısaca Program Çalışma Yöntemi

Öncelikle GridBase adlı abstract bir class oluşturuldu. Bu class grid için gerekli temel özellikleri içeriyor, row ,column, belirli bir noktadan hücreyi alma gibi özellikler. Bir de bu class ana çizme metodunu içeriyor yani hücrelerin özelliğine göre görüntüyü oluşturuyor. Ardından gridbase classı üzerinden 2 adet class oluşturuldu. Bir tanesi labirent için bir tanesi dosyadan yüklemeli olan. Bir robot classı var bu robot classı bir gridbasei parametre olarak alıp Yol bulma işlemleri, hedef ve robotun gridbase üzerine çizimi ile düzenlendi.

5. Akış Şeması




public void Ciz(DrawingContext dc, int gensilik, int cellHeight)
int x = Sutun gensilik
int y = Satir cellHeight
Rect rect = new(x, y, gensilik, cellHeight)
Brush brush = new SolidColorBrush(Renk)
dc.DrawRectangle(brush, null, rect, 5, 5)
_hucreTipi != 0
True
string yazi = $"
(int)_hucreTipi
"
Yazıyı oluştur
var formattedText = new FormattedText
Text = yazi,
Typeface = new Typeface("Arial"),
FontSize = 13,
TextAlignment = TextAlignment.Left,
TextWrapping = TextWrapping.Wrap
Ortayı belirle
var textX = x + (gensilik - formattedText.Bounds.Width) / 2
var textY = y + (cellHeight - formattedText.Bounds.Height) / 2
Çiz
dc.DrawText(Brushes.Black, new Point(textX, textY), formattedText)